智能简历解析系统建设

详细设计说明书

V1.0

猫猫狗狗队

文档修订记录

编号	版本号	修订时间	修订内容	修订人	审核人
1.	V1.0	2023.7.10	初始创建	蔡博凯	谢欣欧

目录

1 引言

1.1 编写目的

详细设计文档的编写目的是提供系统开发过程中的详细设计信息和指南，以便开发团队能够全面理解系统的内部结构、组件之间的关系、数据流程和算法等方面的设计细节。该文档旨在为开发人员提供一个清晰的设计蓝图，使他们能够按照规范和标准进行系统实现。

1.2 文档范围

文档包含模块设计、接口设计、类图设计、数据库设计以及对于核心功能的详细分析与解释。

1.3 阅读对象

详细设计文档的阅读对象主要是项目团队成员，包括开发人员、系统架构师、测试人员和项目管理人员等。

2 模块设计

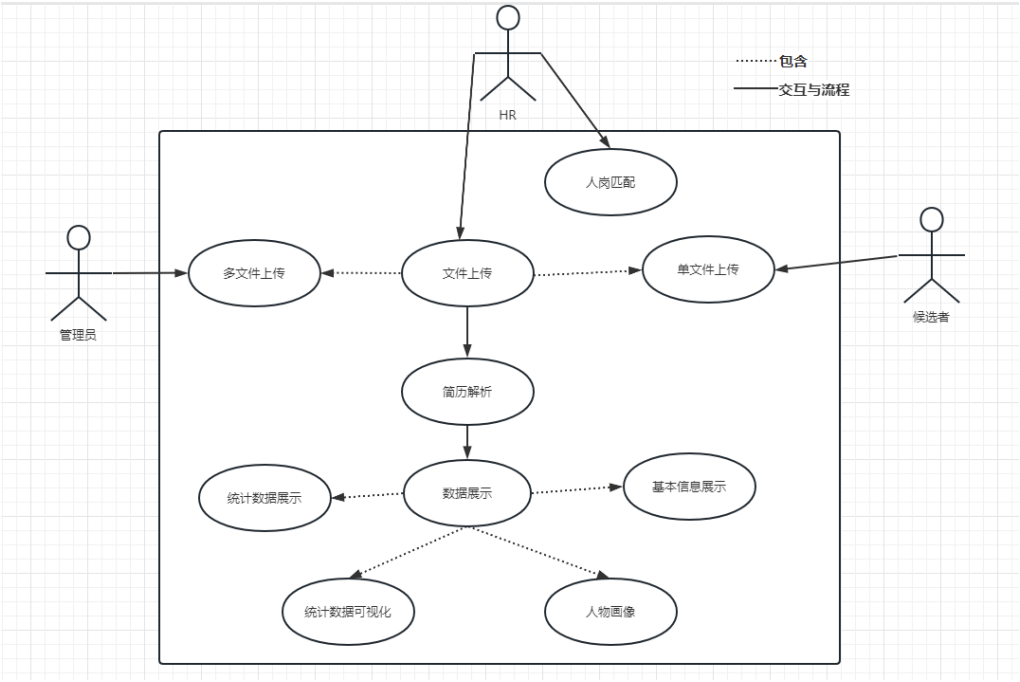
2.1 概述

简历解析系统主要包含四个模块，分别为文件上传模块、简历解析模块、人岗匹配模块以及信息展示模块，其中，文件上传模块包含单文件和多文件上传；信息展示模块包括展示统计数据和基本数据，以及对个人信息进一步分析的人物画像和最后在人岗匹配完成后的匹配简历展示。

2.1.1 模块涉及用户

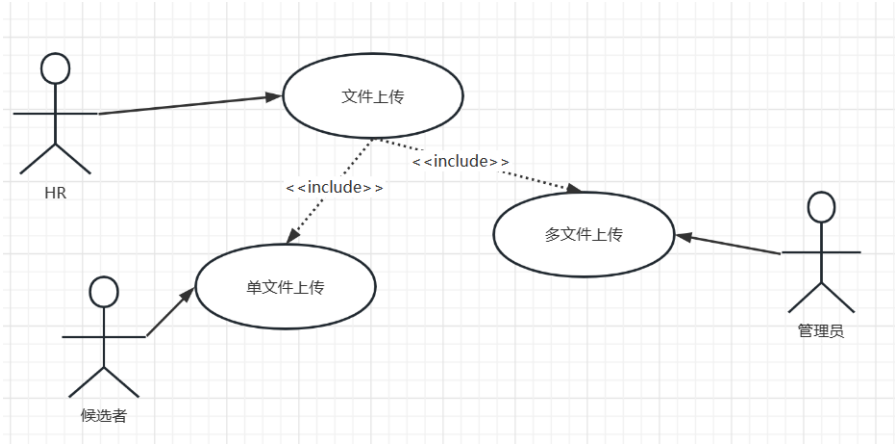
图表 1 模块涉及用户

2.1.2 用例图



图表 2 用例图

2.2 文件上传模块



图表 3 文件上传模块

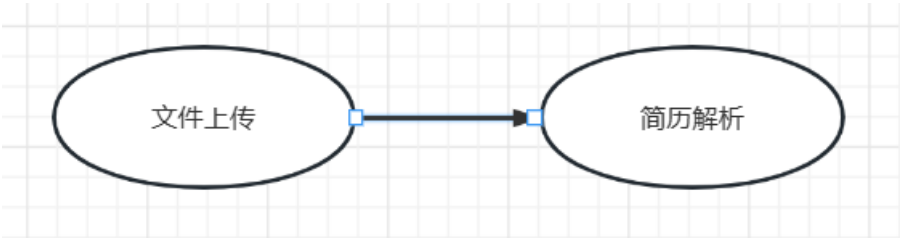
表格 1 单文件上传接口与算法设计

用例名称	单文件上传	
用例对象	HR、候选者	
接口	名称	/singleData/upload
	输入	一份简历文件对象
	输出	简历文件存储路径
算法设计	① 声明临时存储文件路径'uploads/single/' ② 使用在接口中将文件对象存储到对应路径中	

表格 2 多文件上传的接口与算法设计

用例名称	多文件上传	
用例对象	HR、管理员	
接口	名称	/multiData/upload
	输入	多份简历文件对象
	输出	简历文件存储路径
算法设计	① 声明临时存储文件路径'uploads/multi/' ② 使用在接口中将文件对象存储到对应路径中	

2.3 简历解析模块



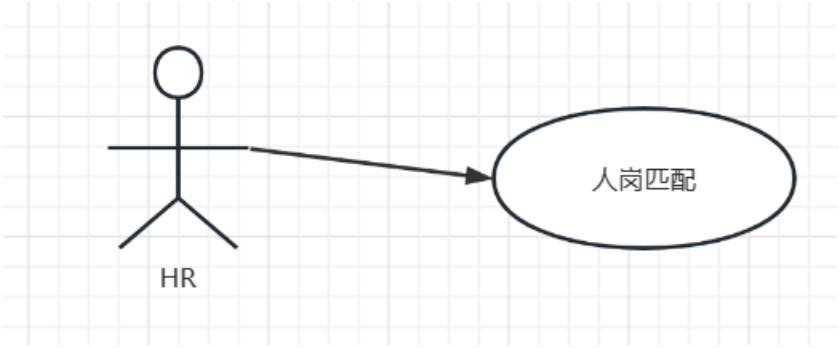
图表 4 简历解析模块

表格 3 简历分析接口与算法设计

用例名称	简历分析	
用例对象	系统	
接口 1	名称	/singleData/dealFile
	输入	uploads/single/
	输出	单份简历解析信息
算法设计 1	① 读取 uploads/single/的文件 ② 通过调用 readFile.py 中的简历解析程序解析文件 ③ 通过调用接口 3 将解析结果更新到数据库中 ④ 解析完文件后自动跳转至基本信息展示页面	
接口 2	名称	/multiData/dealFile
	输入	uploads/multi/
	输出	多份简历解析信息
算法设计 2	① 读取 uploads/multi/的文件 ② 通过调用 readFile.py 中的简历解析程序解析文件 ③ 通过调用接口 4 将解析结果更新到数据库中 ④ 解析完文件后自动跳转至统计信息展示页面	
接口 3	名称	/classifyResume
	输入	单份简历解析信息
	输出	无
接口 4	名称	/classifyMultiResume
	输入	多份简历解析信息
	输出	无

说明	readFile.py 文件会在本文档的核心功能设计中详细说明

2.4 人岗匹配模块

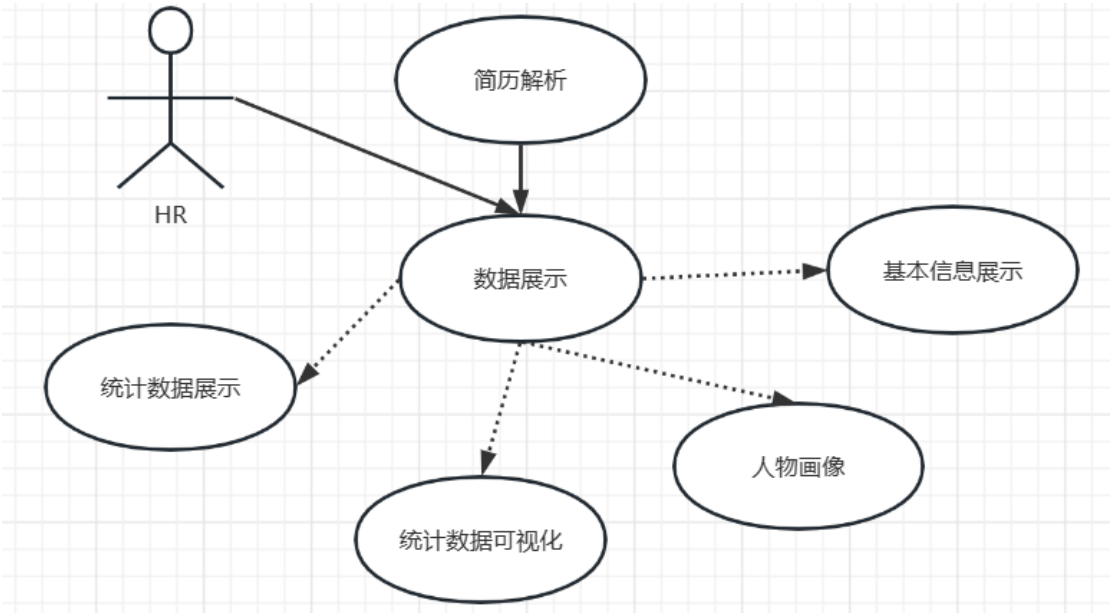


图表 5 人岗匹配模块

表格 4 人岗匹配接口与算法设计

用例名称	人岗匹配	
用例对象	HR	
接口	名称	/jobMatch
	输入	前端提交的岗位信息、数据库中的简历信息
	输出	匹配程度高的简历列表
算法设计	① 前端通过表格收集岗位信息，在 jobMacth.py 中连接数据库，获取相应的简历字段，上述信息会在本文档的类图设计中给出 ② 通过 jobMatch,py 中的 jobMatch 函数计算出某份简历与提供的岗位信息的匹配度 ③ 通过 jobMatch,py 中的 main 函数计算出每一个简历与岗位信息的匹配度，并通过匹配度进行从高到低的排序 ④ 获取前 20 份简历信息，返回给前端界面 ⑤ 前端调用组件展示	
说明	jobMatch.py 文件会在本文档的核心功能设计中详细说明	

2.5 信息展示模块



图表 6 信息展示模块

表格 5 基本信息/人物画像接口与算法设计

用例名称	基本信息展示/人物画像	
用例对象	系统、HR	
接口 1	名称	/showXXXById
	输入	候选者的唯一标识符 id
	输出	标识符为 id 的候选者的 XXX 信息
接口 2	名称	/lightResume
	输入	无
	输出	人物画像分析信息
算法设计	① 后端通过前端返回的信息（id），调用接口 1 得到候选者的 XXX 信息并返回给前端 ② 基本信息直接展示 XXX 字段信息 ③ 人物画像会通过调用接口 2 对 XXX 信息进行分析，并返回分析信息给前端。具体方式是通过观察字段信息来判断，例如判断一个人的英语水平就可以通过提取出的“英语”字段信息来进行 ④ 前端调用组件展示 XXX 信息	

说明	XXX 代表四类简历信息，分别为基本信息，教育信息，工作信息，技能信息，具体字段在需求规格说明书中的 6.1.1 数据需求汇总中介绍了，这里不再赘述

表格 6 统计信息展示接口与算法设计

用例名称	统计信息展示	
用例对象	系统、HR	
接口	名称	/showXXXData
	输入	无
	输出	数据库中所有候选者的 XXX 信息列表
算法设计	① 前端调用上述接口获取全部简历的 XXX 信息 ② 前端调用组件展示 XXX 信息	

表格 7 统计信息可视化接口与算法设计

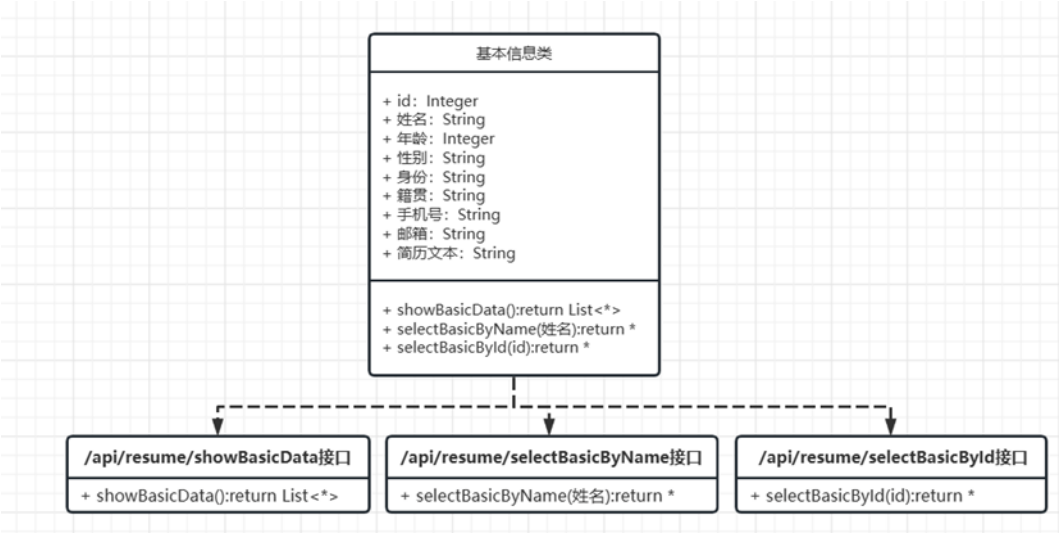
用例名称	统计信息可视化	
用例对象	HR	
接口	名称	/ classifyData
	输入	无
	输出	数据库中所有候选者的统计信息列表
算法设计	① 前端调用上述接口获取全部简历的统计信息，例如分别统计 985/211/其他院校的人数比例 ⑤ 前端调用组件展示统计信息	

3 类图设计

3.1 简历相关类

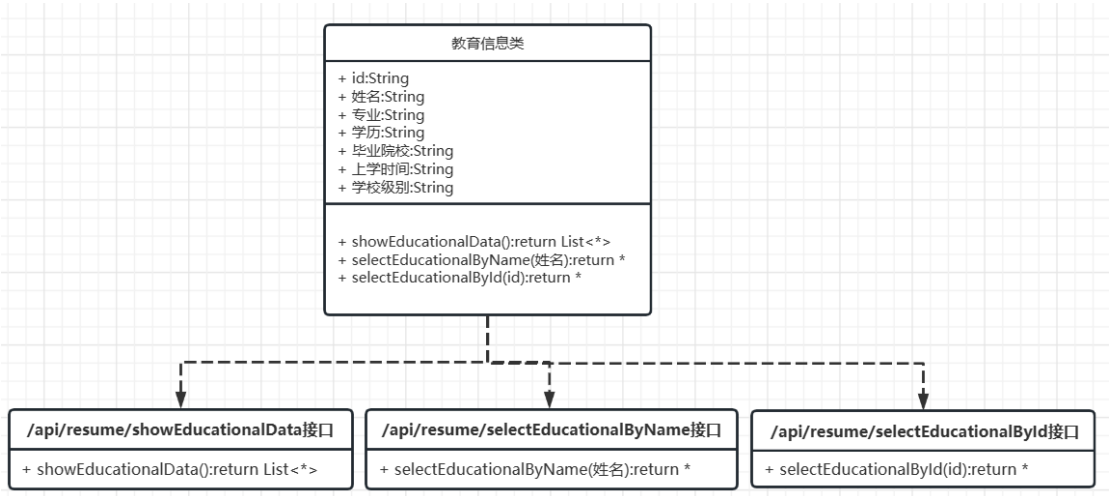
简历相关类包含基本信息类，教育信息类，工作信息类，技能信息类。此外，添加两个简历类，一个用于记录简历上传的全部信息，另一个包含用于人岗匹配时和岗位匹配的相关字段。

3.1.1 基本信息类



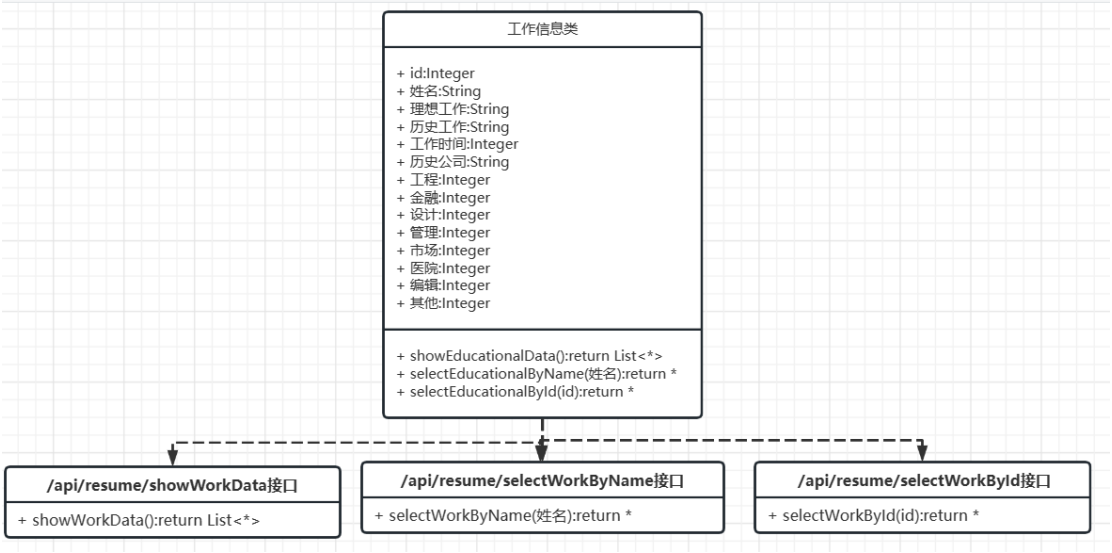
图表 7 基本信息类

3.1.2 教育信息类



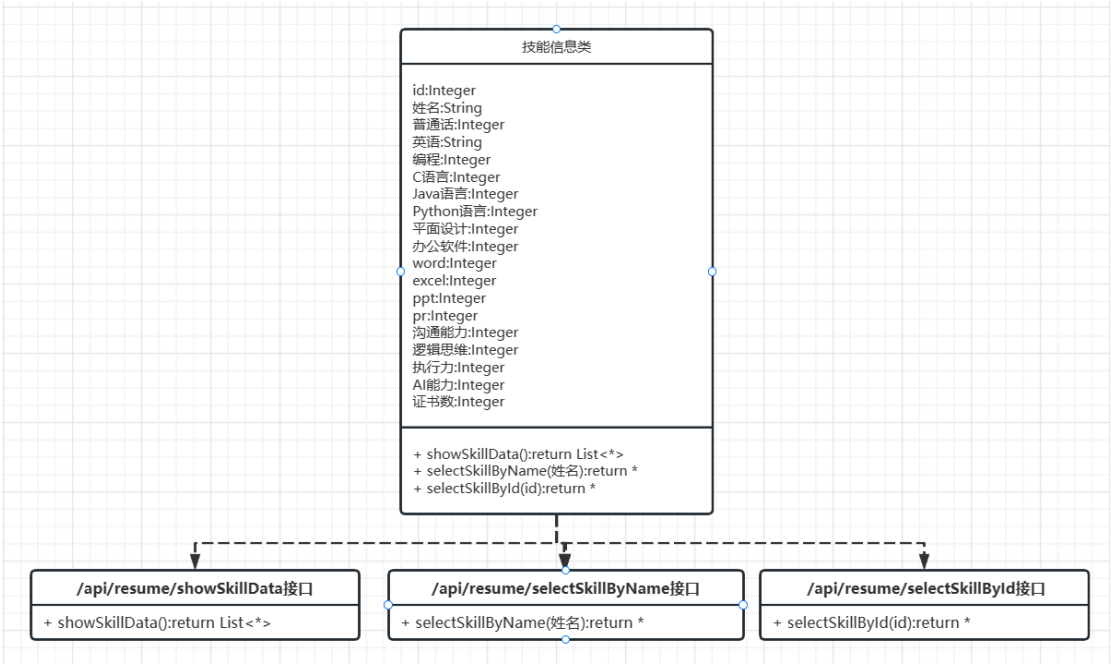
图表 8 教育信息类

3.1.3 工作信息类



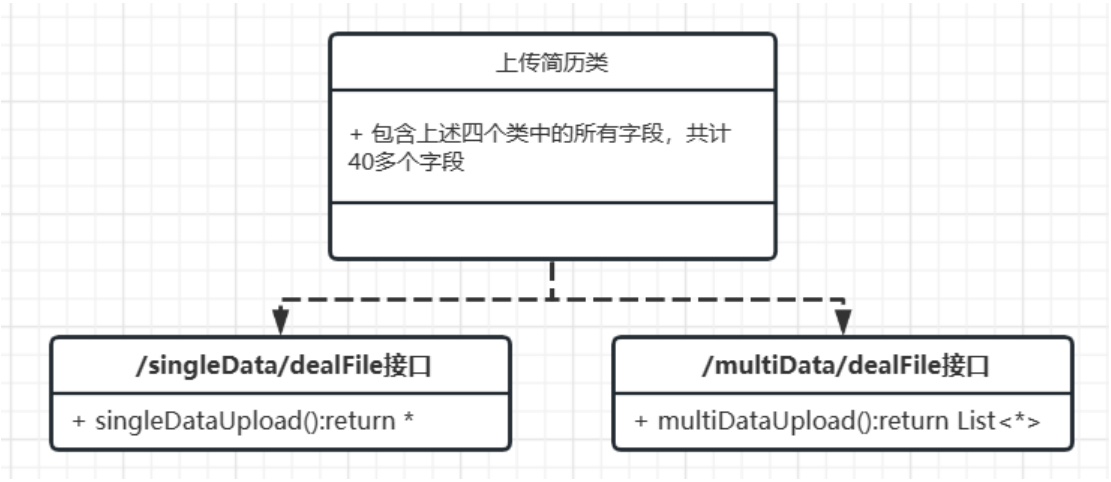
图表 9 工作信息类

3.1.4 技能信息类



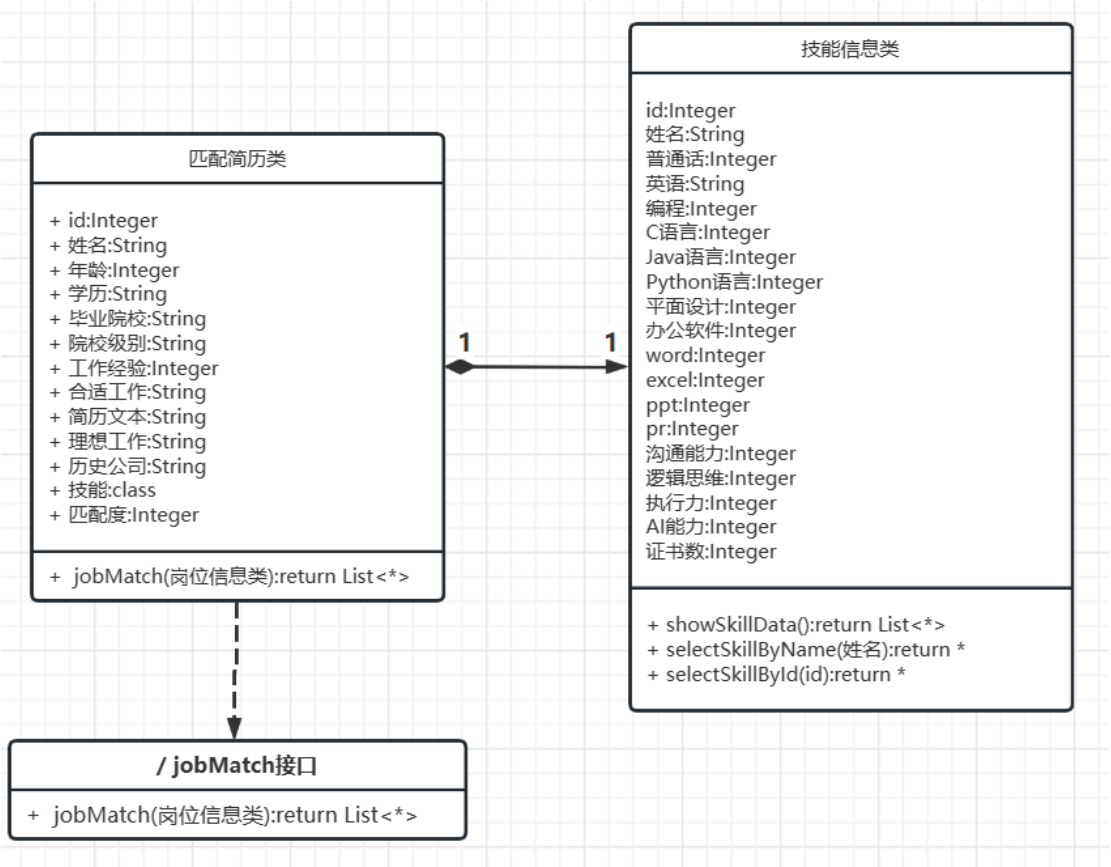
图表 10 技能信息类

3.1.5 上传简历类



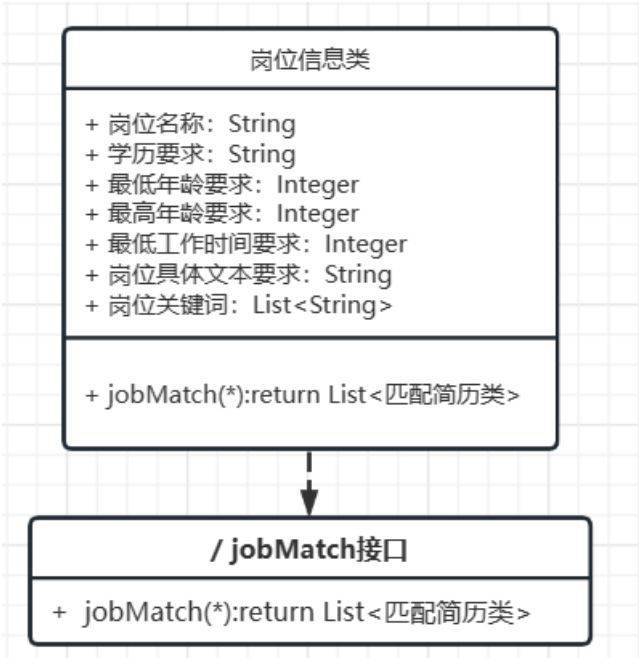
图表 11 上传简历类

3.1.6 匹配简历类



图表 12 匹配简历类

3.2 岗位相关类



图表 13 岗位信息类

4 数据库设计

4.1 基本信息表

数据库 resume_manager				
表名 basic_information				
字段名	字段类型	是否允许为 null	是否为主键	说明
id	int	not null	PK	
name	varchar	not null		
age	int			
gender	varchar			男/女
identity	varchar			党员/群众
origin	varchar			
phone	varchar			

email	varchar			
text	text			待解析的字串

4.2 教育信息表

数据库 resume_manager				
表名 educational_information				
字段名	字段类型	是否允许为 null	是否为主键	说明
id	int	not null	PK	
name	varchar	not null		
major	varchar			专业
education	varchar			学历
university	varchar			毕业院校
schooltime	varchar			上学时间
schoolLevel	varchar			985/211/其他

4.3 工作信息表

数据库 resume_manager				
表名 work_information				
字段名	字段类型	是否允许为 null	是否为主键	说明
id	int	not null	PK	
name	varchar	not null		
targetJob	varchar			理想工作
jobs	varchar			历史工作
workExperience	varchar			工作经验
companies	varchar			历史公司
engineer				判断该简历适

finance design manager market hospital edit elseJob	int			合什么类型的工作，一共有 8 种工作，匹配度 0-100

4.4 技能信息表

数据库 resume_manager				
表名 work_information				
字段名	字段类型	是否允许为 null	是否为主键	说明
id	int	not null	PK	
name	varchar	not null		
mandarin	int			普通话 0-10
english	varchar			四级、六级
computer	int			编程 0-20
graphicDesign	int			平面设计 0-10
officeSoftware	int			办公软件 0-10
reward	int			证书数
c java python word excel ppt pr communicate act	varchar			标签，0/1

logic ai				

5 核心功能设计

5.1 上传文件

规定上传文件的类型有[.doc,.docx,.pdf,.png,.txt]，需要从这些文件中获取简历的文本字符串，如果类型是.doc 则将它化为.docx 处理，如果类型是.docx 通过 win32com 包进行提取，如果类型是.pdf 通过 pdfplumber 包提取，如果类型是.txt 直接打开文件阅读即可，如果类型是.png 调用移动云接口获取。具体代码如下：

```
def request_webimage(imagePath):
    requesturl = '/api/ocr/v1/general'
    try:
        ocr_client = CMSSEcloudOcrClient(accesskey, secretkey, url)
        response =
ocr_client.request_ocr_service_file(requestpath=requesturl, imagepath=
imagePath)
        text = json.loads(response.text)
        if text['errorcode'] == 'AIO.1005':
            return []
        else:
            datalist1 = text['items']
            text = ''
            for data in datalist1:
                text += (data['itemstring'] + '\n')
            return text
    except ValueError as e:
        print(e)

class ReadFile(object):
    '''抽取单个文件的信息 转化为字符串存储'''
    def __init__(self, file_dir):
        self.file_dir = file_dir
        self.text = ''
        listname = os.path.splitext(self.file_dir)
        if listname[1] in [".doc", ".docx",".pdf",".png",".txt"]:
            try:
                if listname[1] == ".doc":
```

```
                    self.__doc2docx()
                self.text = self.__extract_text()
            except Exception as e:
                print(e)
                return

    def __doc2docx(self):
        '''doc 转 docx'''
        word = win32com.client.Dispatch('Word.Application')
        word.Visible = 0 # 后台运行
        word.DisplayAlerts = 0 # 不显示，不警告
        doc = word.Documents.Open(self.file_dir)
        new_path = os.path.splitext(self.file_dir)[0] + '.docx'
        doc.SaveAs(new_path,16)
        doc.Close()
        word.Quit()
        os.remove(self.file_dir)
        self.file_dir = new_path
        return self.file_dir
    def __extract_text(self):
        '''抽取文本内容'''
        text = ''
        listname = os.path.splitext(self.file_dir)
        if listname[1] == '.docx':
            document = ZipFile(self.file_dir)
            xml = document.read("word/document.xml")
            wordObj = BeautifulSoup(xml.decode("utf-8"),features='xml')
            #用来删除 fallback 标签，实现内容去重
            fallback_tags = wordObj.find_all("mc:Fallback")
            for fallback_tag in fallback_tags:
                fallback_tag.decompose()
            texts = wordObj.findAll("w:t")
            for t in texts:
                text += (t.text + '\n')
        elif listname[1] == '.pdf':
            with pdfplumber.open(self.file_dir) as pdf:
                for page in pdf.pages:
                    text += page.extract_text() if page.extract_text()
else ""
        elif listname[1] == '.png':
            text = request_webimage(self.file_dir)
        elif listname[1] == '.txt':
            with open(self.file_dir,'r',encoding='utf-8') as f:
                text = f.read()
```

```
        return text
```

5.2 解析文件

通过解析上传文件获取的字符串获得下列主要信息。

5.2.1 提取姓名

提取姓名通过 xml 的格式进行字符串初步处理，具体方法是将每一行加上分隔符，然后删除空行。将处理后的文本放入 LAC 模型中通过 NER 功能获取姓名信息，具体代码如下：

```
    # 提取姓名
    def searchName(self,sentences: str) -> list:
        # 进行 xml 形式处理
        sentences = '<w:t>' + sentences
        sentences = sentences.replace("\n", "</w:t><w:t>")
        #删除空格
        sentences = re.sub('</w:t><w:t> </w:t><w:t>','',sentences)
         # 装载 LAC 模型
        user_name_list = []
        lac_result = self.lac.run(sentences)
        for index, lac_label in enumerate(lac_result[1]):
            if lac_label == "PER":
                user_name_list.append(lac_result[0][index])

        # 将列表元素作为 OrderedDict 的键创建一个新字典
        d = OrderedDict.fromkeys(user_name_list)
        # 将 OrderedDict 的键转换为列表，即为去重后的列表
        user_name_list = list(d.keys())

        #对新提取的结果进行必要的过滤

        #删除实际内容小于 1 的字符串
        lst = [item for item in user_name_list if
re.search(r'[^><\s]{2,}', item)]
        #删除实际内容包括英文的字符串
        res_lst = [item for item in lst if not re.search(r'[a-zA-Z0-9]',
item)]

        res = ''
```

```
        if (len(res_lst) == 0): res = unknown
        else:
            result = re.findall(r'[\u4e00-\u9fff]+', res_lst[0])
            result = ''.join(result)
            res = result.strip()
        return res
```

5.2.2 提取年龄

先通过正则表达式找出包含“年龄：”字段的行，在后面可以找到年龄，如果找不到的话继续寻找包含“岁”字段的行，在前面可以找到年龄，如果找不到的话找到简历中的最小年时间，将其于 2023 作差后加 1 即可获知年龄。如果找到的年龄在 18 到 50 岁之间，视为正常；反之视为不正常，并将年龄修改为 20 岁。具体代码如下：

```
# 提取年龄
    def searchAge(self,datatext):
        datatext = re.sub(r'\s','',datatext)
        if re.search(r'年龄[:： ]{0,1}([1-5]\d{1})',datatext):
            age = re.findall(r'年龄[:： ]{0,1}([1-5]\d{1})',datatext)[0]
            return int(age)
        elif re.search(r'[1-5]\d{1}岁',datatext):
            age = re.findall(r'[1-5]\d{1}岁',datatext)[0]
            age = re.sub(r'[岁\s]',"",age)
            return int(age)
        else:
            births = re.findall(r"(?:19|20)\d{2}", datatext)
            births = list(map(int,births))
            births.sort()
            age = 2023 - births[0] + 1 if len(births) > 0 else 2002
            if (age < 18 or age > 50) and re.search(r'(?:18|[2-
5]\d)\D',datatext):
                age = re.findall(r'(?:18|[2-5]\d)\D',datatext)[0]
                age = re.sub(r'\D','',age)
            else:
                age = 20
            return int(age)
```

5.2.3 提取学历

先通过 jieba 包里的 cut 方法分词，然后查找简历文本中是否有“职业技术学

院”、“专科学校”、“职业学院”等字样，如果有的话，说明学历为大专。遍历所有的词，如果词语中包含“中专”，“本科”这些字样，即可获知学历。最后在通过 paddlenlp 包里的 information_extraction 模块抽取文本中的学历信息。具体代码如下：

```
# 提取学历
    def searchDegree(self,text):
        schema = ['学位']
        text = re.sub('\n','',text)
        education_background_result = ""
        education_background = ['中专','大专','本科','本科生','本科专业','
硕士','硕士专业','硕士生','博士','博士生']
        tex = list(jieba.cut(text))
        if text.find('职业技术学院')!=-1 or text.find('职业学院')!=-1 or
text.find('专科学校')!=-1:
            education_background_result = '大专'
        for j in range(len(education_background)):
            if education_background[j] in tex:
                education_background_result =
education_background[j][0:2]
        if education_background_result == '':
            func = Taskflow("information_extraction", schema=schema,
task_path=self.information_extraction)
            result = func(text)
            if len(result) > 0:
                if '学位' in result[0]:
                    education_background_result = result[0]['学位
'][0]['text'][0:2]
                else:
                    education_background_result = '大专'
            else:
                education_background_result = '大专'

        return education_background_result
```

5.2.4 提取毕业院校

先通过教育部提供的大学名录.txt 文件查询简历中是否有对应的学校，如果找不到就通过 paddlenlp 包里的 information_extraction 模块抽取文本中的毕业院校信息。具体代码如下：

```
def extract_university_name(sentence, universities):
```

```
    escaped_universities = [re.escape(university) for university in
universities]
    # 为大学名称创建正则表达式模式
    pattern =
r"(?<![.])({})(?![.])".format("|".join(escaped_universities))
    # 从输入句子中查找匹配的大学名称
    matches = re.findall(pattern, sentence)
    return matches

# 提取毕业院校
    def searchSchool(self,text):
        schema = ['毕业院校']
        text = re.sub('\n','',text)
        education_graduate_school_result = ""
        result = extract_university_name(text,
self.chinese_universities)
        if len(result) > 0:
            education_graduate_school_result = result[0]
        else:
            func = Taskflow("information_extraction", schema=schema,
task_path=self.information_extraction)
            result = func(text)
            if len(result) > 0:
                if '毕业院校' in result[0]:
                    education_graduate_school_result = result[0]['毕业院
校'][0]['text']
                else:
                    education_graduate_school_result = unknown
            else:
                education_graduate_school_result = unknown
        return education_graduate_school_result
```

5.2.5 提取工作经验

工作时间截止到 2023.4，首先提取出文本中所有可能的工作年份，对它进行标准化，将“至今”改为“2023.4”，根据已提取到的年龄信息，对年份进行分割，主要目的在于剔除在学校的年份，随后在统计工作年龄是进一步剔除掉实习和兼职阶段的年龄。具体代码如下

```
# 提取工作时间
    def searchWorkYears(self,text,age,degree):
        '''提取工作时长'''
        text = re.sub(r'\s','',text)
```

```
        target1 = r'(20\d{2}(?:\.(?:0\d|1[0-2]|\d)?)?[-到–一—
–]+20\d{2}(?:\.(?:0\d|1[0-2]|\d)?)?)'
        target2 = r'(20\d{2}(?:年(?:0\d|1[0-2]|\d)?月?)?[-到–一—
–]+20\d{2}(?:年(?:0\d|1[0-2]|\d)?月?)?)'
        target3 = r'\S{0,30}?(?:实习|兼职)\S{0,30}?'
        target5 = r'\S{0,30}?(?:大学|学院|学校|教育背景|校园|社长|主席|社
团)\S{0,30}?'
        target6 = r'20\d{2}[年.]{1}(?:0\d|1[0-2]|\d)?月?[-到–一—]*[至
今]{1,2}'
        target7 = r'((?:0\d|1[0-2]|\d)\.20\d{2})'
        target8 = r'(20\d{2}(?:\.(?:0\d|1[0-
2]|\d)?)?)(20\d{2}(?:\.(?:0\d|1[0-2]|\d)?)?)'
        if re.search(target7,text):
            tmp = re.findall(target7,text)
            for line in tmp:
                line1 = re.split(r'[.]',line)[1] + '.' +
re.split(r'[.]',line)[0]
                text = re.sub(line,line1,text)
            text = re.sub(target8,r'\1' + '-' + r'\2',text)
        if re.search(target6,text):
            tmp = re.findall(target6,text)[0]
            tmp = re.sub(r'[-到–一—]*[至今]{1,2}','-2023.4',tmp)
            text = re.sub(target6,tmp,text)
        times = re.findall(target1,text) + re.findall(target2,text)
        times = sorted(list(set(times)))
        border = 2023
        if degree == '本科':
            border -= (age - 22)
        elif degree == '硕士':
            border -= (age - 25)
        elif degree == '博士':
            border -= (age - 28)
        elif degree == '大专':
            border -= (age - 21)
        elif degree == '中专':
            border -= (age - 18)
        months = 0
        acc = []
        useless = []
        for i in range(len(times)):
            line = times[i]
            year1,year2,month1,month2 = 0,0,0,0
            line1 = re.sub(r'年','.',line)
            line1 = re.sub(r'月','',line1)
```

```
            yearS,yearE = re.findall(r'20\d{2}(?:\.(?:0\d|1[0-
2]|\d)?)?',line1)
            if yearS[-1] == '.':
                yearS = yearS[:-1]
            if yearE[-1] == '.':
                yearE = yearE[:-1]
            if '.' in yearS:
                year1,month1 = yearS.split('.')
            if '.' in yearE:
                year2,month2 = yearE.split('.')
            tmp = (int(year2) - int(year1)) * 12 + (int(month2) -
int(month1))
            if int(year2) <= border:
                useless.append(line)
                continue
            elif int(year1) > border:
                if re.search(target3 + line,text) or re.search(line +
target3,text):
                    useless.append(line)
                    continue
            else:
                if re.search(target5 + line,text) or re.search(line +
target5,text):
                    useless.append(line)
                    continue
                if re.search(target3 + line,text) or re.search(line +
target3,text):
                    useless.append(line)
                    continue
            break
        for line in times:
            if line not in useless:
                acc.append(line)
        yearS = yearE = ''
        if len(acc) > 0:
            line1 = re.sub(r'年','.',acc[0])
            line1 = re.sub(r'月','',line1)
            line2 = re.sub(r'年','.',acc[-1])
            line2 = re.sub(r'月','',line2)
            yearS = re.findall(r'20\d{2}(?:\.(?:0\d|1[0-
2]|\d)?)?',line1)[0]
            yearE = re.findall(r'20\d{2}(?:\.(?:0\d|1[0-
2]|\d)?)?',line2)[1]
        year1,year2,month1,month2 = 0,0,0,0
```

```
        if '.' in yearS:
            year1,month1 = yearS.split('.')
        if '.' in yearE:
            year2,month2 = yearE.split('.')
        if month1 == '':
            month1 = 0
        if month2 == '':
            month2 = 0
        months = (int(year2) - int(year1)) * 12 + (int(month2) -
int(month1))
        if months % 12 == 0:
            return int(months / 12)
        else:
            return int(months / 12) + 1
```

5.3 人岗匹配

通过岗位名称，工作经验，年龄要求匹配对应的简历。这里有两种角度，一种是候选者角度，即为一份简历匹配多种岗位，一种是 HR 角度，即为一个岗位找出与之适配度高的简历，在简历解析系统中采用的是后者，但前者也做了实现，具体代码如下：

候选者角度

```
jobs = {
    "产品运营":{
        "name":["产品运营"],
        "work":2,
        "degree":"无",
        "age":18,
    },
    "平面设计师":{
        "name":["平面设计师","平面设计"],
        "work":1,
        "degree":"大专",
        "age":18,
    },
    "财务":{
        "name":["财务","会计"],
        "work":0,
        "degree":"本科",
        "age":18,
    },
```

```
    "市场营销":{
        "name":["市场营销"],
        "work":10,
        "degree":"本科",
        "age":18,
    },
    "项目主管":{
        "name":["项目主管","总监","市场管理"],
        "work":3,
        "degree":"本科",
        "age":18,
    },
    "开发工程师":{
        "name":["开发工程师"],
        "work":3,
        "degree":"本科",
        "age":18,
    },
    "文员":{
        "name":["文员","助理"],
        "work":1,
        "degree":"大专",
        "age":25,
    },
    "电商运营":{
        "name":["电商运营"],
        "work":2,
        "degree":"无",
        "age":18,
    },
    "人力资源管理":{
        "name":["人力资源管理"],
        "work":2,
        "degree":"大专",
        "age":18,
    },
    "风控专员":{
        "name":["金融","国际贸易"],
        "work":5,
        "degree":"硕士",
        "age":18,
    },
}
degrees = {"无":-1,"中专":0,"大专":1,"本科":2,"硕士":3,"博士":4}
```

```
def jobMatch(resume:Extractor):
    result = []
    for key, value in jobs.items():
        if value["work"] > resume.workTime:
            continue
        if value["age"] > resume.age:
            continue
        if degrees[value["degree"]] > degrees[resume.degree]:
            continue
        for name in value["name"]:
            if re.search(name,resume.text):
                result.append(key)
                continue
    return "、".join(result)
```

HR 角度

```
def jobMatch(job:Job,resumer:Resumer):
    '''岗位匹配'''
    score = 0
    if re.search(job.jobName,resumer.text):
        score += 100
    if resumer.education == '博士':
        score += 10
    elif resumer.education == '硕士':
        score += 8
    elif resumer.education == '本科':
        score += 5
    if resumer.schoolLevel == '985':
        score += 10
    elif resumer.schoolLevel == '211':
        score += 8
    e = {'中专':0,'大专':1,'本科':2,'硕士':3,'博士':4}
    sim = 0
    if resumer.education not in e:
        resumer.education = '本科'
    if e[job.education] > e[resumer.education]:
        return 0
    if job.workMinTime > resumer.workExperience:
        return 0
    if job.ageMax < resumer.age:
        return 0
    if job.ageMin > resumer.age:
        return 0
    for j in resumer.targetJob:
```

```
        sim = max(sim,Levenshtein.jaro(j,job.jobName)) # 是0-1吗
    sim = int(50 * sim)
    score += sim
    count = 0
    for k in job.keyWords:
        if re.search('(?i)' + k,resumer.text):
            count += 1
    key = int(30 * (count / len(job.keyWords))) if len(job.keyWords) > 0
else 0
    score += key
    return int(score * 0.5)
```

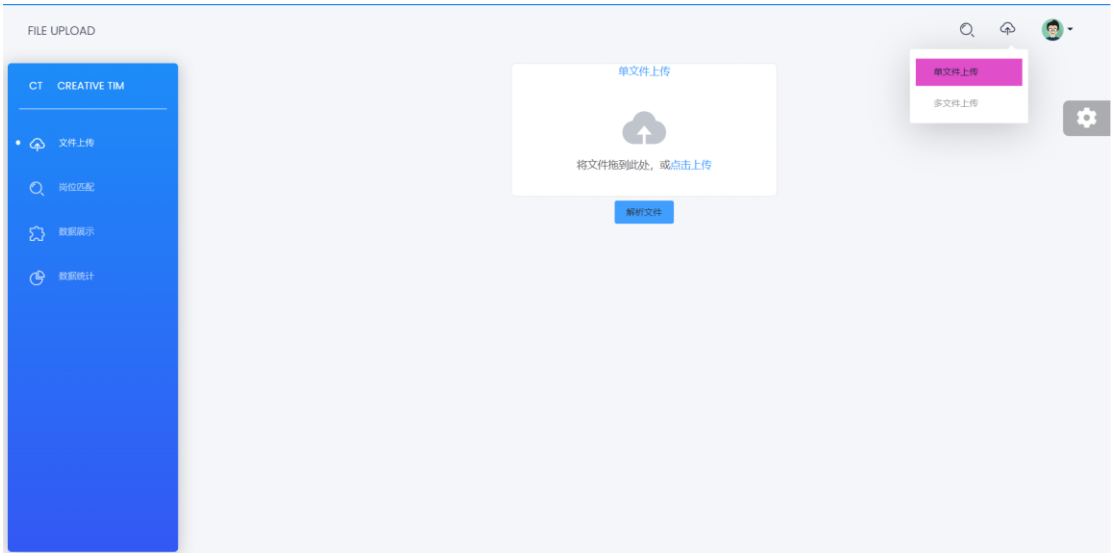
5.4 训练数据正确率

姓名	年龄	学历	毕业院校	工作经验	岗位匹配
95%	100%	95%	91%	87%	90%

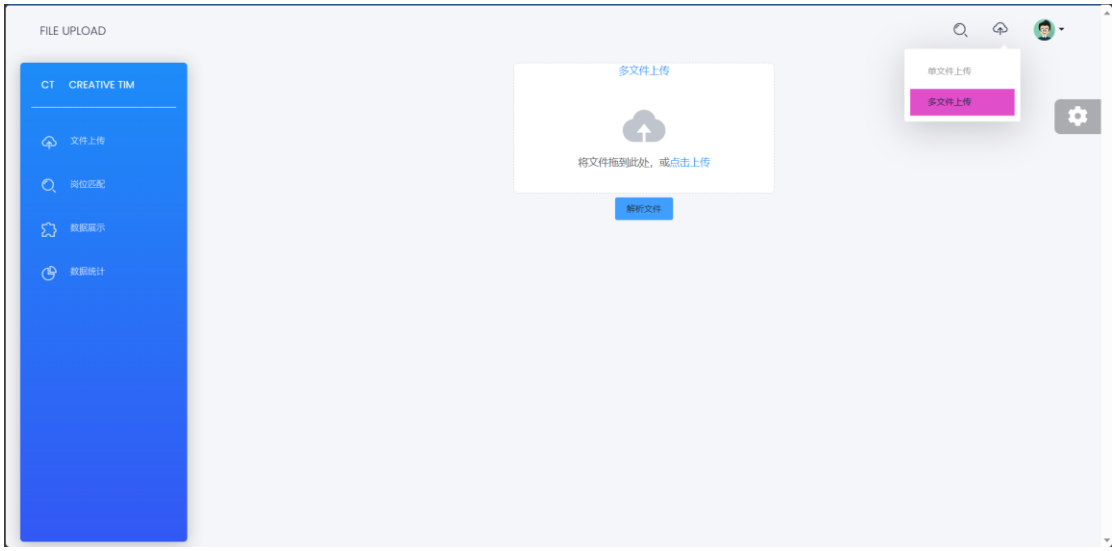
6 界面详细设计

6.1 文件上传界面

6.1.1 界面



图表 14 单文件上传



图表 15 多文件上传

6.1.2 代码

```
1.     <template>
2.       <div class="content">
3.         <el-upload class="upload-
demo" drag :action="backendEndpoint" :multiple="true" :before-
upload="handleBeforeUpload"
4.           :on-success="handleSuccess">
5.           <i class="el-icon-upload"></i>
6.           <div class="el-upload__text">将文件拖到此处，或<em>点击上
传</em></div>
7.           <div class="el-upload__tip" slot="tip">支持上传多人简历文
件</div>
8.         </el-upload>
9.         <div class="upload-status" v-if="uploadCount > 0">已成功
上传 {{ uploadCount }} 个文件</div>
10.        <div class="upload-status" v-
if="uploadFailedFiles.length > 0">
11.          以下文件上传失败：
12.          <ul>
13.            <li v-
for="file in uploadFailedFiles" :key="file.name">{{ file.name }}<
/li>
14.          </ul>
```

```
      </div>
    </div>
  </template>

  <script>
  import axios from 'axios';

  export default {
    data() {
      return {
        backendEndpoint: "http://localhost:8181/api/file/upload",
        uploadCount: 0,
        uploadFailedFiles: []
      };
    },
    methods: {
      handleBeforeUpload(file) {
        const formData = new FormData();
        formData.append('kind', 'multiData');
        formData.append('file', file);

        axios.post(this.backendEndpoint, formData, {
          headers: {
            'Content-Type': 'multipart/form-data'
          }
        })
          .then(response => {
            this.uploadCount++;
            console.log('文件上传成功:', file, '响应:', response);
          })
          .catch(error => {
            this.uploadFailedFiles.push(file);
            console.error('上传文件错误:', file, '错误信息:', error);
          })
          .finally(() => {
```

```
            this.showUploadStatus();
          });

        return false; // 阻止默认的上传行为
      },
      handleSuccess(response, file) {
        console.log('文件上传成功:', file, '响应:', response);
      },
      showUploadStatus() {
        setTimeout(() => {
          this.uploadCount = 0;
          this.uploadFailedFiles = [];
        }, 5000);
      }
    }
  };
  </script>

  <style>
  .with-background {
    background-image: url('../assets/img/beijing.jpg');
    background-repeat: no-repeat;
    background-size: cover;
  }

  .container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
  }

  .d-flex {
    display: flex;
  }

  .justify-content-center {
    justify-content: center;
```

```
88.     }
89.
90.     .upload-status {
91.       margin-top: 10px;
92.       font-weight: bold;
93.     }
94.     </style>
```

6.1.3 代码设计

在<template>标签中，定义了一个包含文件上传功能的界面。其中，使用了<el-upload>组件来实现文件上传的功能。<el-upload>组件通过设置不同的属性来配置上传的行为，例如 drag 表示支持拖拽上传，:action 指定了上传的后端接口地址，:multiple 表示支持多文件上传等。

在<script>标签中，通过 import 语句引入了 axios 库，用于发送 HTTP 请求。然后，通过 export default 导出一个 Vue 组件对象。在该组件的 data 属性中定义了需要用到的数据，包括 backendEndpoint 表示后端接口地址，uploadCount 表示已成功上传的文件数量，uploadFailedFiles 表示上传失败的文件数组。

在组件中定义了一些方法，包括 handleBeforeUpload 用于处理文件上传前的操作，handleSuccess 用于处理上传成功后的操作，以及 showUploadStatus 用于显示上传状态信息。

在<style>标签中，定义了一些样式规则，例如设置背景图片、设置容器居中显示、设置上传状态的样式等。

根据代码的设计，该文件上传界面使用了 Vue.js 框架，同时还依赖了 Element UI 组件库（通过<el-upload>组件）。在上传文件时，通过调用 axios.post 方法向后端发送 HTTP 请求，使用 FormData 对象来构造表单数据，以支持文件的上传。

6.2 基本信息展示

6.2.1 界面

图表 16 基本信息展示

6.2.2 代码

```
1.     <template>
2.         <div class="content with-background">
3.             <div id="app">
4.                 <v-app>
5.                     <v-main>
6.                         <v-data-
table :headers="headers" :items="desserts" :items-per-
page="15" class="elevation-1">
7.                             <template v-slot:item.操作
="{ item }">
8.                                 <v-
btn @click="viewDetails(item.id)">详情</v-btn>
9.                             </template>
10.                        </v-data-table>
11.                    </v-main>
12.                </v-app>
13.            </div>
14.        </div>
```

```
15.    </template>
16.
17.    <script>
18.    import axios from 'axios';
19.
20.    export default {
21.
22.        data() {
23.            return {
24.                headers: [
25.                    { text: '序号
', align: 'start', value: 'id' },
26.                    { text: '姓名
', value: 'name', sortable: false },
27.                    { text: '年龄', value: 'age' },
28.                    { text: '工作年限
', value: 'workExperience' },
29.                    { text: '最高学历
', value: 'mostEducation', sortable: false },
30.                    { text: '毕业院校
', value: 'university', sortable: false },
31.                    { text: '操作', value: '操作
', sortable: false },
32.                ],
33.                desserts: [],
34.            };
35.        },
36.        mounted() {
37.            this.fetchData();
38.        },
39.        methods: {
40.            fetchData() {
41.                axios
42.                    .get('http://localhost:8181/api/resume/showB
asicData')
43.                    .then((response) => {
44.                        // 处理基本数据的响应
45.                        const basicData = response.data.data;
```

```
46.
47.                         // 调用其他接口获取最高学历和毕业院校数据
48.                         return axios
49.                             .get('http://localhost:8181/api/resume/showEducationalData')
50.                             .then((response) => {
51.                                 // 处理最高学历和毕业院校数据的响应
52.                                 const educationalData = response.data.data;
53.
54.                                 // 调用另一个接口获取工作年限数据
55.                                 return axios
56.                                     .get('http://localhost:8181/api/resume/showWorkData')
57.                                     .then((response) => {
58.                                         // 处理工作年限数据的响应
59.                                         const workData = response.data.data;
60.
61.                                         // 组合数据
62.                                         const processedData = basicData.map((item, index) => ({
63.                                             id: item.id,
64.                                             name: item.name,
65.                                             age: item.age,
66.                                             mostEducation: educationalData[index].education,
67.                                             university: educationalData[index].university,
68.                                             workExperience: workData[index].workExperience,
69.                                         }));
70.
71.                                         this.desserts = processedData;
72.                                         console.log(this.desserts);
73.                                     })
```

```
                                        .catch((error) => {
                                            console.error(error);
                                        });
                                })
                                .catch((error) => {
                                    console.error(error);
                                });
                        })
                        .catch((error) => {
                            console.error(error);
                        });
                },

                viewDetails(id) {
                    this.$router.push(`/details/${id}`);
                },
            },
        };
    </script>

    <style>
    .with-background {
        background-image: url('../assets/img/beijing.jpg');
        background-repeat: no-repeat;
        background-size: cover;
    }
    </style>

```

6.2.3 代码设计

以下是对代码的设计解释：

在<template>标签中，定义了一个包含信息展示功能的界面。其中使用了<v-data-table>组件来展示表格数据，该组件通过设置不同的属性来配置表格的列和行，例如 headers 定义了表格的表头，items 表示表格的数据项，items-per-page 指定了每页显示的数据项数量。在组件中还定义了一个<template>标签，用于自定

义表格中的操作列。

在<script>标签中，通过 import 语句引入了 axios 库，用于发送 HTTP 请求。然后，通过 export default 导出一个 Vue 组件对象。在该组件的 data 属性中定义了需要用到的数据，包括 headers 表示表格的表头信息，desserts 表示表格的数据项。在组件的 mounted 生命周期钩子中，调用了 fetchData 方法来获取数据并初始化表格。在组件中定义了 fetchData 方法，用于发送 HTTP 请求获取基本数据、最高学历和毕业院校数据以及工作年限数据，并进行数据的处理和组合，最终将处理后的数据赋值给 desserts。其中使用了多个嵌套的 axios.get 方法来处理多个异步请求，确保数据的顺序和完整性。viewDetails 方法用于处理点击操作列中的"详情"按钮的事件，通过 this.$router.push 方法跳转到详情页面，并将对应的 id 作为参数传递。在<style>标签中，定义了一个样式规则，设置了背景图片。根据代码的设计，该信息展示界面使用了 Vue.js 框架，并结合了 Vuetify UI 库（通过<v-data-table>和<v-btn>组件）来实现数据的展示和操作。在组件中，通过使用 axios 库发送 HTTP 请求获取后端数据，并通过嵌套的异步请求来处理多个接口的响应数据。通过这种方式，可以将不同接口返回的数据组合成需要的格式，并在表格中展示出来。

6.3 统计信息展示

6.3.1 界面

图表 17 统计信息展示

图表 18 统计信息可视化

6.3.2 代码

```
<template>
  <div class="content with-background">
    <div class="row">
      <div class="col-1"></div>
      <div class="col-10">
        <div class="card-container">
          <Card shadow>
            <div class="resume-section-header">
              <div class="small-line-left-archieve"></div>
              <h4 class="section-word">学历信息</h4>
              <div class="small-line-right-archieve"></div>
            </div>
            <div>
              <el-row :gutter="20">
                <el-col :span="4">
                  <div>
                    <el-statistic
                      group-separator=","
                      :value="value1"
                      :title="title1"
                    ></el-statistic>
                  </div>
                </el-col>
```

```
                    <el-col :span="4">
                      <div>
                        <el-statistic
                          group-separator=","
                          :value="value2"
                          :title="title2"
                        ></el-statistic>
                      </div>
                    </el-col>
                    <el-col :span="4">
                      <div>
                        <el-statistic
                          group-separator=","
                          decimal-separator="."
                          :value="value3"
                          :title="title3"
                        >
                          <template slot="prefix">
                            <i class="el-icon-s-
flag" style="color: red"></i>
                          </template>
                          <template slot="suffix">
                            <i class="el-icon-s-
flag" style="color: blue"></i>
                          </template>
                        </el-statistic>
                      </div>
                    </el-col>
                    <el-col :span="4">
                      <div>
                        <el-
statistic :value="value4" :title="title4">
                          <template slot="suffix">
                            <span @click="toggleLike" class="lik
e">
                              <i
                                class="el-icon-star-on"
                                style="color: red"
```

```
                                v-show="like"
                              ></i>
                              <i
                                class="el-icon-star-off"
                                v-show="!like"
                              ></i>
                            </span>
                          </template>
                        </el-statistic>
                      </div>
                    </el-col>
                    <el-col :span="4">
                      <div>
                        <el-statistic
                          group-separator=","
                          :value="value5"
                          :title="title5"
                        ></el-statistic>
                      </div>
                    </el-col>
                  </el-row>
                </div>
                <div class="resume-section-header">
                  <div class="small-line-left-archieve"></div>
                  <h4 class="section-word">其他信息</h4>
                  <div class="small-line-right-archieve"></div>
                </div>
                <v-tabs v-model="activeTab">
                  <v-tab @click="activeTab = 0">毕业院校</v-tab>
                  <v-tab @click="activeTab = 1">工作时长</v-tab>
                  <v-tab @click="activeTab = 2">年龄</v-tab>
                </v-tabs>

                <div class="card-body">
                  <ve-donut-chart
                    v-if="activeTab === 0"
                    :data="chartData1"
                    :settings="chartSettings1"
```

```
                    />
                    <ve-donut-chart
                      v-if="activeTab === 1"
                      :data="chartData2"
                    />
                    <ve-pie-chart
                      v-if="activeTab === 2"
                      :data="chartData3"
                      :settings="chartSettings3"
                    />
                  </div>
                </Card>
              </div>
            </div>
          </div>
        </template>
```

6.3.3 代码设计

在<template>标签中，定义了一个包含学历信息和其他信息的信息展示界面。界面分为两个部分，学历信息和其他信息。学历信息部分使用了 el-statistic 组件展示各个学历类型的数量，使用了 el-row 和 el-col 布局组件来排列多个学历类型的统计信息。其他信息部分使用了 v-tabs 组件来切换显示不同的图表，根据 activeTab 的值决定显示哪个图表。

在<script>标签中，通过 import 语句引入了 Card 组件和 axios 库，用于展示卡片和发送 HTTP 请求。然后，通过 export default 导出一个 Vue 组件对象。在该组件的 components 属性中注册了 Card 组件，以便在模板中使用。

在组件的 data 属性中定义了需要用到的数据。其中包括各个学历类型的标题（title1 到 title5），以及学历类型的数量（value1 到 value5）。activeTab 表示当前激活的选项卡，chartData1、chartData2 和 chartData3 分别表示三个图表的数据，chartSettings1 和 chartSettings3 表示图表的设置。在组件的 created 生命周期钩子中，调用了 fetchChartData 方法来获取数据并初始化图表。fetchChartData 方法通过发送 HTTP 请求获取后端 API 返回的数据，将各个学历类型的数量赋值给相

应的 value 变量，更新 chartData1、chartData2 和 chartData3 的数据，以便在图表中展示。在处理响应数据时，使用了对象的 keys 和 values 方法来获取键和值的数组。toggleLike 方法用于切换喜欢按钮的状态，当点击按钮时，通过修改 like 变量的值来控制显示不同的图标。

在<style>标签中，定义了一些样式规则，包括标题样式、背景图片样式等。根据代码的设计，该信息展示界面使用了 Vue.js 框架，并结合了 Element UI 库（通过 el-statistic、el-row、el-col 和 v-tabs 组件）来实现数据的展示和交互。在组件中，通过使用 axios 库发送 HTTP 请求获取后端数据，并将返回的数据分别赋值给相应的变量，以便在模板中展示和使用。

6.4 人物画像

6.4.1 界面

图表 19 人物画像

6.4.2 代码

```
1.     <template>
2.         <div class="content">
3.             <div class="topUser">
4.
5.                 <img v-if="gender === '男
'" src="../assets/img/anime3.png" width="70" height="70">
```

```
6.                  <img v-
else src="../assets/img/anime6.png" width="70" height="70">
7.                  <div class="topUser_right">
8.                      <div class="topUser_right_top">
9.                          <div class="name">{{ userName }}</div>
10.                         <div class="namebq">{{ university }}</di
v>
11.                         <div class="namebq">{{ capacity }}</div>
12.                     </div>
13.                     <div class="topUser_right_bottom">
14.                         <div class="item" style="margin-
left: 0;">
15.                             <i class="el-icon-user-solid"></i>
16.                             <span> {{ year }}</span>
17.                         </div>
18.                         <div class="item">
19.                             <i class="el-icon-remove"></i>
20.                             <span> {{ gender }}</span>
21.                         </div>
22.                         <div class="item">
23.                             <i class="el-icon-phone"></i>
24.                             <span> {{ number }}</span>
25.                         </div>
26.                         <div class="item">
27.                             <i class="el-icon-message"></i>
28.                             <span> {{ emit }}</span>
29.                         </div>
30.                         <div class="item">
31.                             <i class="el-icon-message"></i>
32.                             <span> {{ experience }}</span>
33.                         </div>
34.                         <div class="item">
35.                             <i class="el-icon-suitcase-1"></i>
36.                             <span> {{ education }}</span>
37.                         </div>
38.                     </div>
39.                 </div>
40.             </div>
```

```
41.             <el-tabs class="table-tab" v-model="activeName">
42.                 <el-tab-pane label="中文简历" name="first">
43.                     <div class="userInfo">
44.                         <div class="title">
45.                             <div class="info">基本信
息 <div class="pos"></div>
46.                             </div>
47.                             <el-button type="success">查看返回数据
</el-button>
48.                         </div>
49.                         <div class="lists">
50.                             <ul>
51.                                 <li><span class="label">姓
名: </span> {{ userName }}</li>
52.                                 <li><span class="label">手
机: </span> {{ number }}</li>
53.                                 <li><span class="label">籍
贯: </span> {{ native }}</li>
54.                                 <li><span class="label">专
业:</span> {{ speciality }}</li>
55.                                 <li><span class="label">工作时
长:</span> {{ Working }}</li>
56.                                 <li><span class="label">政治面
貌:</span> {{ outlook }}</li>
57.                             </ul>
58.                             <ul style="margin-left: 300px;">
59.                                 <li><span class="label">年
龄: </span> {{ year }}</li>
60.                                 <li><span class="label">邮
箱: </span> {{ emit }}</li>
61.                                 <li><span class="label">性
别: </span> {{ gender }}</li>
62.                                 <li><span class="label">学
历:</span> {{ education }}</li>
63.                                 <li><span class="label">毕业院
校:</span> {{ institutions }}</li>
64.                                 <li><span class="label">毕业院校
类别:</span> {{ category }}</li>
```

```
65.                                 </ul>
66.                             </div>
67.                         </div>
68.                         <div>
69.                             <div class="title">
70.                                 <div class="info">基本信
息 <div class="pos"></div>
71.                                 </div>
72.                             </div>
73.                             <div class="nameusdate">
74.                                 <div>{{ institutions }}</div>
75.                                 <div class="date">{{ date }}</div>
76.                             </div>
77.                             <div class="lists">
78.                                 <ul>
79.                                     <li><span class="label">开始时
间: </span> {{ startDate }}</li>
80.                                     <li><span class="label">结束时
间: </span> {{ ennDate }}</li>
81.                                     <li><span class="label">地
点: </span> {{ gender }}</li>
82.                                     <li><span class="label">学
位:</span> {{ education }}</li>
83.                                     <li><span class="label">专
业:</span> {{ speciality }}</li>
84.                                     <li><span class="label">学校级
别:</span> {{ category }}</li>
85.                                 </ul>
86.                             </div>
87.                         </div>
88.                         <div>
89.                             <div class="title">
90.                                 <div class="info">补充信
息 <div class="pos"></div>
91.                                 </div>
92.                             </div>
93.                             <div class="lists">
94.                                 <ul>
```

```
95.                                 <li><span class="label">普通话水平: </span> {{ pth }}</li>
96.                                 <li><span class="label">英语水平: </span>     {{ elevel }}</li>
97.                                 <li><span class="label">编程语言: </span> {{ comcap }}</li>
98.                                 <li><span class="label">平面设计能力:</span> {{ designcap }}</li>
99.                                 <li><span class="label">办公软件掌握:</span> {{ mastercap }}</li>
100.                                <li><span class="label">剪辑能力:</span> {{ prcap }}</li>
101.                                <li><span class="label">沟通\逻辑思维\执行\AI 能力:</span> {{ othercap }}</li>
102.                            </ul>
103.                        </div>
104.                    </div>
105.                    <div class="hr"></div>
106.                </el-tab-pane>
107.                <el-tab-pane label="候选人画像" name="second">
108.                    <div class="title">
109.                        <div class="info">简历亮点 <div class="pos"></div>
110.                        </div>
111.                        <el-button type="success">查看返回数据</el-button>
112.                    </div>
113.                    <div class="Highlights">
114.                        <div class="titLabel">
115.                            <div class="image">
116.                                <i class="el-icon-thumb"></i>
117.                            </div>
118.                            <p style="color: black;">简历亮点 x5 </p>
119.                        </div>
120.                        <ul>
121.                            <li>{{ titLabel1 }}</li>
122.                            <li>{{ titLabel2 }}</li>
```

```
123.                             <li>{{ titLabel3 }}</li>
124.                             <li>{{ titLabel4 }}</li>
125.                             <li>{{ titLabel5 }}</li>
126.                         </ul>
127.                     </div>
128.
129.                     <div class="title">
130.                         <div class="info">候选人标
签 <div class="pos"></div>
131.                         </div>
132.                     </div>
133.                     <div style="margin-left: 10px;">
134.                         <div class="titles">基本标签</div>
135.                         <div class="labels">
136.                             <el-tooltip v-
for="item in tooltipList" :content="item.content" placement="top"
>
137.                                 <el-
button type="primary" plain round>{{ item.label }}</el-button>
138.                             </el-tooltip>
139.                         </div>
140.                     </div>
141.                     <div style="margin-left: 10px;">
142.                         <div class="titles">教育背景标签</div>
143.                         <div class="labels">
144.                             <el-tooltip v-
for="item in tooltipList2" :content="item.content" placement="top
">
145.                                 <el-
button type="warning" plain round>{{ item.label }}</el-button>
146.                             </el-tooltip>
147.                         </div>
148.                     </div>
149.
150.                     <div style="margin-left: 10px;">
151.                         <div class="titles">技能标签</div>
152.                         <div class="labels">
```

```
                            <el-tooltip v-for="item in tooltipList3" :content="item.content" placement="top">
                                <el-button style="background: #5bda92;border-color:#5bda92 ;" type="success" round>{{ item.label }}</el-button>
                            </el-tooltip>
                        </div>
                    </div>


                    <div class="title" style="margin-top:  10px;">
                        <div class="info">职位甄别 <div class="pos"></div>
                        </div>
                    </div>
                    <ve-radar-chart :data="chartData" :legend="legend" :settings="settings" />
                    <div class="title" style="margin-top:  10px;">
                        <div class="info">能力展示 <div class="pos"></div>
                        </div>
                    </div>
                    <div ref="chart2" style="width: 600px; height: 600px; margin-left: 35%"></div>


                </el-tab-pane>


            </el-tabs>
        </div>
    </template>

```

6.4.3 代码设计

<template> 标签定义了组件的模板部分。整个组件被包裹在一个名为 "content" 的 元素中。 定义了一个顶部用户信息的容器。在顶部用户信息中，根据 gender 的值判断展示不同的头像图片。 定义了用户信息的右侧部分。{{ userName }}、{{ university }}、{{ capacity }} 等变量通过双花括号语法插入到对应的位置，用于显示用户信息。<el-tabs> 是一个基于 Element UI 库的标签页组件，通过 v-model="activeName" 绑定当前激活的标签页名称。<el-tab-pane> 标签表示一个标签页，通过 label 属性设置标签页的显示文本，name 属性设置标签页的唯一标识。第一个标签页展示中文简历信息，包括基本信息、教育背景和补充信息等。第二个标签页展示候选人画像，包括简历亮点、候选人标签、职位甄别和能力展示等。 定义了一个具有 "chart2" 引用的 元素，用于展示能力展示图表。样式设置了宽度为 600px，高度为 600px，左边距为 35%。

6.5 人岗匹配

6.5.1 界面

图表 20 人岗匹配

6.5.2 代码

```
1.     <template>
2.       <div class="content" style="min-height: 100%">
3.         <el-row :gutter=15 style="height: 100%">
4.           <el-col :span="10">
5.             <el-card>
6.               <el-row>
7.                 <h3>职位信息</h3>
8.               </el-row>
9.               <el-form ref="form" :model="form" label-
width="80px" class="t-left">
10.                <el-form-item label="岗位名称" title="jobname">
11.                  <el-input v-model="form.jobname" placeholder="
请输入岗位名称"></el-input>
12.                </el-form-item>
13.                <el-form-item label="学历要求" title="education">
14.                  <el-select v-
model="form.education" placeholder="请选择学历" class="t-left">
15.                    <el-option label="中专" value="中专"></el-
option>
16.                    <el-option label="大专" value="大专"></el-
option>
17.                    <el-option label="本科" value="本科"></el-
option>
18.                    <el-option label="硕士" value="硕士"></el-
option>
19.                    <el-option label="博士" value="博士"></el-
option>
20.                  </el-select>
21.                </el-form-item>
22.                <el-form-item label="年龄" title="Min(Max)age">
23.                  <el-row class="age">
24.                    <el-input type="number" v-
model="form.minAge" placeholder="最小年龄"></el-input><span>-
</span><el-input
25.                      type="number" max="60" v-
model="form.maxAge" placeholder="最大年龄"></el-input>
```

```
26.                   </el-row>
27.                   <!-- <el-select v-
model="form.age" placeholder="请选择年龄段" class="t-left">
28.                     <el-option label="20~25" value="20~25"></el-
option>
29.                     <el-option label="26~30" value="26~30"></el-
option>
30.                     <el-option label="31~35" value="31~35"></el-
option>
31.                     <el-option label="36~40" value="36~40"></el-
option>
32.                     <el-option label="41~45" value="41~45"></el-
option>
33.                     <el-option label="46~50" value="46~50"></el-
option>
34.                   </el-select> -->
35.                 </el-form-item>
36.                 <el-form-item label="工作经验
" title="Min(Max)workTime">
37.                   <el-select v-
model="form.workTime" placeholder="请选择工作经验" class="t-left">
38.                     <el-option label="1 年" value="1"></el-
option>
39.                     <el-option label="2 年" value="2"></el-
option>
40.                     <el-option label="3 年" value="3"></el-
option>
41.                     <el-option label="4 年" value="4"></el-
option>
42.                     <el-option label="5 年" value="5"></el-
option>
43.                   </el-select>
44.                 </el-form-item>
45.                 <el-form-item label="岗位要求">
46.                   <el-input type="textarea" :rows="3" v-
model="form.desc" placeholder="请输入岗位要求" class="t-left"></el-
input>
47.                 </el-form-item>
```

```
48.                 <el-form-item class="submit">
49.                   <el-button type="primary" @click="onSubmit">提
交</el-button>
50.                 </el-form-item>
51.               </el-form>
52.             </el-card>
53.           </el-col>
54.           <el-col :span="14" class="card-list">
55.             <el-card class="card-item" v-
for="item in matchcont" :key="item">
56.               <el-row>
57.                 <h4>{{ item.target }}</h4>
58.               </el-row>
59.               <el-rate v-model="value" disabled text-
color="#ff9900" :score-template="'{{ item.score }}'">
60.
61.               </el-rate>
62.               <el-row><span>{{ item.name }}</span><el-
divider direction="vertical"></el-divider><span>{{ item.education
63.               }}</span><el-divider direction="vertical"></el-
divider><span>{{ item.workExperience
64.     }}年工作经验</span></el-row>
65.               <el-row><span>{{ item.company }}</span><el-
divider direction="vertical"></el-divider><span>{{ item.target
66.               }}</span></el-row>
67.               <el-row><span>{{ item.university }}</span><el-
divider direction="vertical"></el-divider><span>{{ item.major
68.               }}</span><el-divider direction="vertical"></el-
divider><span>{{ item.schoolLevel }}院校</span></el-row>
69.               <el-row>
70.                 <el-tag v-if="item.skill.act !== 0">行动力</el-
tag>
71.                 <el-tag v-if="item.skill.ai !== 0">人工智能</el-
tag>
72.                 <el-tag v-if="item.skill.c !== 0">C 语言</el-tag>
73.                 <el-tag v-if="item.skill.communicate !== 0">沟通
能力</el-tag>
```

```
74.                 <el-tag v-if="item.skill.computer !== 0">计算机
</el-tag>
75.                 <el-tag v-if="item.skill.english === '四级
' || item.skill.english === '六级'">{{ item.skill.english }}</el-
tag>
76.                 <el-tag v-if="item.skill.excel !== 0">excel</el-
tag>
77.                 <el-tag v-if="item.skill.graphicDesign !== 0">平
面设计能力</el-tag>
78.                 <el-tag v-if="item.skill.mandarin !== 0">普通话
</el-tag>
79.                 <el-tag v-if="item.skill.officeSoftware !== 0">
掌握办公软件</el-tag>
80.                 <el-tag v-if="item.skill.ppt !== 0">ppt</el-tag>
81.                 <el-tag v-if="item.skill.pr !== 0">剪辑</el-tag>
82.                 <el-tag v-
if="item.skill.python !== 0">python</el-tag>
83.                 <el-tag v-
if="item.skill.reward !== 0">{{ item.skill.reward }}个荣誉证书</el-
tag>
84.                 <el-tag v-if="item.skill.word !== 0">word</el-
tag>
85.               </el-row>
86.             </el-card>
87.           </el-col>
88.         </el-row>
89.       </div>
90.     </template>
91.
92.     <script>
93.     import axios from 'axios';
94.
95.     export default {
96.       data() {
97.         return {
98.           form: {
99.             jobname: "",
100.            education: "",
```

```
            minAge: 0,
            maxAge: 0,
            workTime: '',
            desc: "",
          },
          value: 5,
          matchcont: []
        };
      },
      methods: {
        onSubmit() {
          console.log(this.form)
          axios
            .post("http://localhost:5000/jobMatch", this.form, {
              headers: { "Content-Type": "application/x-www-form-urlencoded" }
            })
            .then(response => {
              this.matchcont = response.data
              console.log(this.matchcont[0].skill)


            })
            .catch(error => {
              console.error(error)
            })
        }
      }
    };
    </script>

    <style>
    .t-left {
      text-align: left !important;
    }

    h3,
    h4 {
```

```
    font-weight: 700;
    color: #444 !important;
  }

  h4 {
    margin-bottom: 0 !important;
  }

  .el-form-item label {
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
  }

  .submit .el-form-item__content {
    margin-left: 0 !important;
    text-align: center;
  }

  .el-tag {
    margin: 5px 10px 5px 0;
  }

  .card-item {
    margin-bottom: 10px;
  }

  .card-list {
    max-height: 540px;
    overflow: auto;
  }

  .age {
    display: flex;
  }

  .age span {
    margin: 0 10px;
```

```
176.    }
177.    </style>
```

6.5.3 代码设计

1.模板部分（<template>）：

使用 Element UI 框架的组件来构建页面布局和表单元素。页面布局使用<el-row>和<el-col>来创建两列布局。第一列（左侧）包含一个卡片（<el-card>），显示职位信息的表单。表单包含输入框、下拉框、数字输入框和文本域等元素，用于填写职位相关的信息。提交按钮（<el-button>）用于提交表单数据。第二列（右侧）是一个卡片列表，使用 v-for 指令根据数据项动态生成卡片。卡片显示匹配的职位信息，包括职位名称、评分、教育要求、工作经验等。在卡片底部显示各种技能标签，根据职位要求动态展示。

2.脚本部分（<script>）：

引入 Axios 库用于发送 HTTP 请求。定义了组件的数据对象，包括表单数据和匹配结果的数组。定义了 onSubmit 方法，在提交按钮点击时触发。onSubmit 方法使用 Axios 发送 POST 请求到 http://localhost:5000/jobMatch 接口，并将表单数据作为请求参数。请求成功后，将返回的匹配结果赋值给 matchcont 数组，并输出到控制台。

3.样式部分（<style>）：

样式部分定义了一些基本的样式规则，包括文本对齐、字体样式等。为表单元素和卡片列表设置了一些特定的样式，例如文本溢出省略、最大高度和滚动等。